

CSM RESEARCH: TESTBED DEVELOPMENT

Ronnie E. Gilliant†

NASA Langley Research Center

INTRODUCTION

The Computational Structural Mechanics(CSM) Activity at Langley Research Center is developing methods for structural analysis on modern computers. To facilitate that research effort, a Testbed Development environment is being constructed. It is the purpose of the Testbed to insulate researchers from differences in the computer operating systems of modern computer systems and permit concentrated effort on the analytical problem rather than the analytical tools required to solve that problem.

While modern computers enable the solution of larger problems of increasing complexity, they do so at a cost. Distributed computer environments, vector processing hardware and multiple processors dominate the current computer environment and threaten to overwhelm future analysis software. The systems software for current computers becomes more complex in order to manage the increasing complexity. The applications developer is caught between conflicting goals. They must take advantage of the computing power of new computer systems while maintaining a stable software development system. The CSM Testbed is being developed to address this problem for the computational structural analysis research community.

The Langley CSM activity was initiated in October 1984. This paper describes the current directions for the Testbed Development Team of the Langley CSM activity.

† Mathematician, Structural Mechanics Branch, Structures and Dynamics Division.

LANGLEY CSM PROGRAM

Parallel Structural Methods Research - Olaf O. Storaasli

Testbed Development - Ronnie Gillian

Methods and Applications Studies - Norm Knight

LANGLEY CSM PROGRAM

The CSM activity at Langley is divided into three activities, parallel structural methods, Testbed development, and methods research. This talk will describe the current activity in the Testbed Development area.

The Testbed Development area is located between the advanced computation area dominated by a computer science team, and the methods research dominated by structural engineers. It is an attempt to bridge the gap between these two disciplines and bring both groups closer to the realities imposed by implementation.

CSM TESTBED DEFINITION

A computer program for developing and evaluating advanced analysis methods and software architecture for a new generation of computers.

CSM TESTBED DEFINITION

The CSM Testbed is a computer program for developing and evaluating advanced analysis methods and software architecture for a new generation of computers. This definition emphasizes several different areas.

First, the Testbed is software. It is a computer program consisting of over 100,000 lines of FORTRAN source code with low level I/O in either "C" or assembly language.

The program will be used in both the development mode for new software methodology, and the evaluation mode for structural concepts. This dual purpose requires the careful balance of the competing constraints of efficiency and generality.

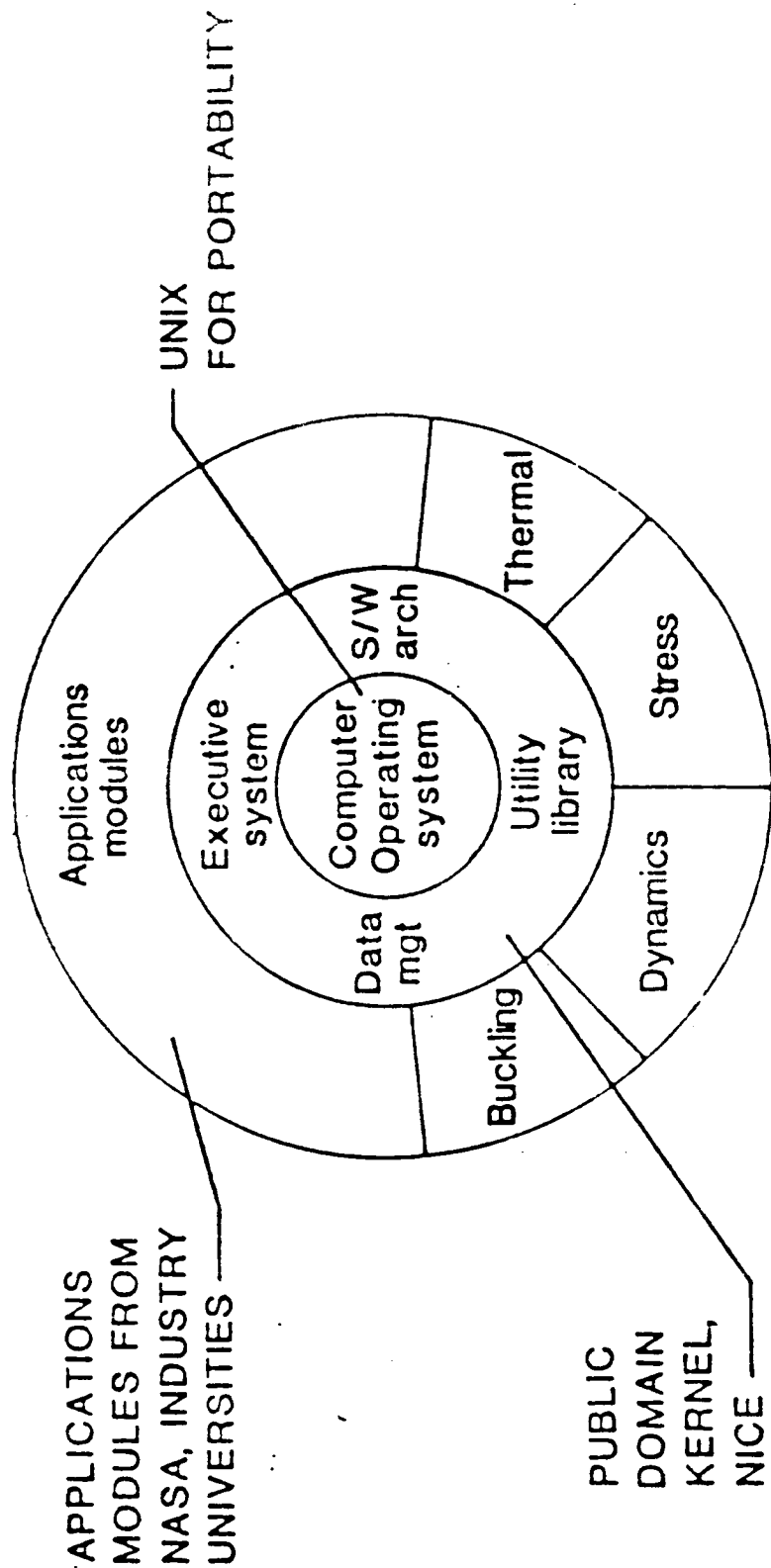
Our CSM Testbed provides the mechanism for research into both analysis methods and the software architecture required to support those analysis methods. As the analysis methods change, the software architecture must change and the Testbed serves as a framework under which differing architectures and methods may be investigated.

Lastly, the common thread of advanced computers runs throughout the Testbed. The development of advanced computer hardware is continuing to accelerate. New architecture computers implementing extremely sophisticated hardware and software concepts that "revolutionize" scientific computing appear two or three times a month. We must be able to evaluate and use these computer enhancements within the CSM area quickly.

COMMON GENERIC SOFTWARE SYSTEM

LARC TESTBED

- PUBLIC DOMAIN SOFTWARE
- MECHANISM FOR TECHNOLOGY TRANSFER
- INDUSTRY, UNIVERSITY, NASA INVOLVEMENT



COMMON GENERIC SOFTWARE SYSTEM

The CSM Testbed as currently implemented at Langley provides a mechanism for technology transfer through the use of public domain software and involves industry, universities, and NASA in a total integrated program.

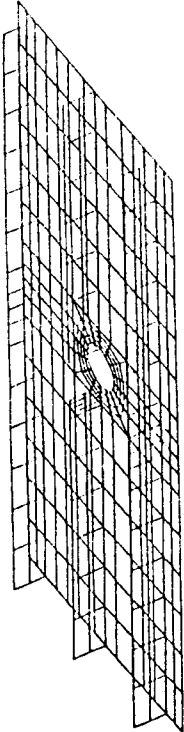
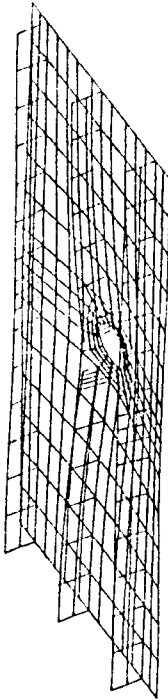
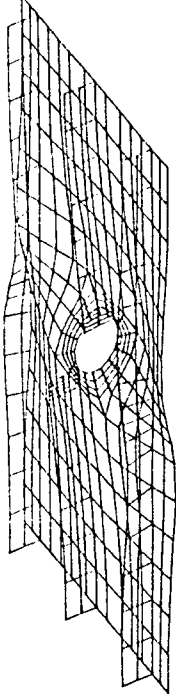
The best way to think of the CSM Testbed is as a group of concentric shells. The inner shell represents the computer operating system. We have concentrated on the UNIX operating system for the Testbed to provide portability. UNIX is the operating system used by all supercomputers and most workstations and is readily available for the range of computers in between. While all versions of UNIX are not directly compatible, they do provide most of the same commands and extensions due to the influence of AT&T. Although we will continue to support the Testbed in VAX VMS environment, new work will be first implemented under UNIX.

The second shell, the executive system shell, consists of the command line interpreter, data management facility, and utility library. This facility was provided initially by the NICE computer code from Lockheed and it has been upgraded to its present state through additional contracts with Lockheed and in-house effort. This shell serves to isolate the outer shell from the computer operating system.

The outer shell consists of the application processors that really do the structural analysis tasks involved in CSM. This outer shell was initially provided by the SPAR system processors and has been augmented through grants, the Lockheed contract and in-house effort.

This Testbed forms a basis on which we can continue the development of structural analysis methodology and apply it to current problems in a way previously not available.

BUCKLING ANALYSIS USING THE CSM TESTBED

PROCESSOR	FUNCTION	RESULT
TAB CSM1★ LAU★ ELD AUS	MODEL GENERATION	 UNDEFORMED SHAPE
RSEQ★ E EKS TOPO K	FORM AND ASSEMBLE STIFFNESS MATRICES	 LINEAR SOLUTION
INV SSOL	FACTOR AND SOLVE	
GSF PSF	STRESS RECOVERY	
KG EIG	BUCKLING ANALYSIS	 BUCKLING MODE

★NEW PROCESSORS DEVELOPED FOR THE TESTBED

BUCKLING ANALYSIS USING THE CSM TESTBED

An example of a problem the Testbed can be used to solve is the buckling analysis of the CSM Focus problem. The desired result of the analysis can be seen in the lower right corner plot of the buckling mode. Intermediate results are used to check the progress of the solution. For example, the undeformed shape in the upper right corner is used to check the output of the model generation phase.

The model generation phase of the solution involves the processors TAB, CSM1, LAU, ELD, and AUS. The processors CSM1 and LAU were generated in-house in order to routinely generate this type of structure and to describe the laminations of a composite material.

The linear solution, center right, results from the continued analysis phases required to form and assemble the stiffness matrices with processors RSEQ, E, EKS, TOPO, and K. The RSEQ renumbering processor was developed to reduce the computational overhead associated with the system of equations. Factoring and solving these matrices are accomplished with the processors INV and SSOL. Actual stresses are recovered through the processors GSF and PSF.

The KG and ELG processors are used to complete the buckling analysis computations. The graphic output in this case was provided by PLTA and PLTB.

The software model implemented here is one of a series of computational processors controlled through the use of a higher level language with data management provided through an efficient data base manager.

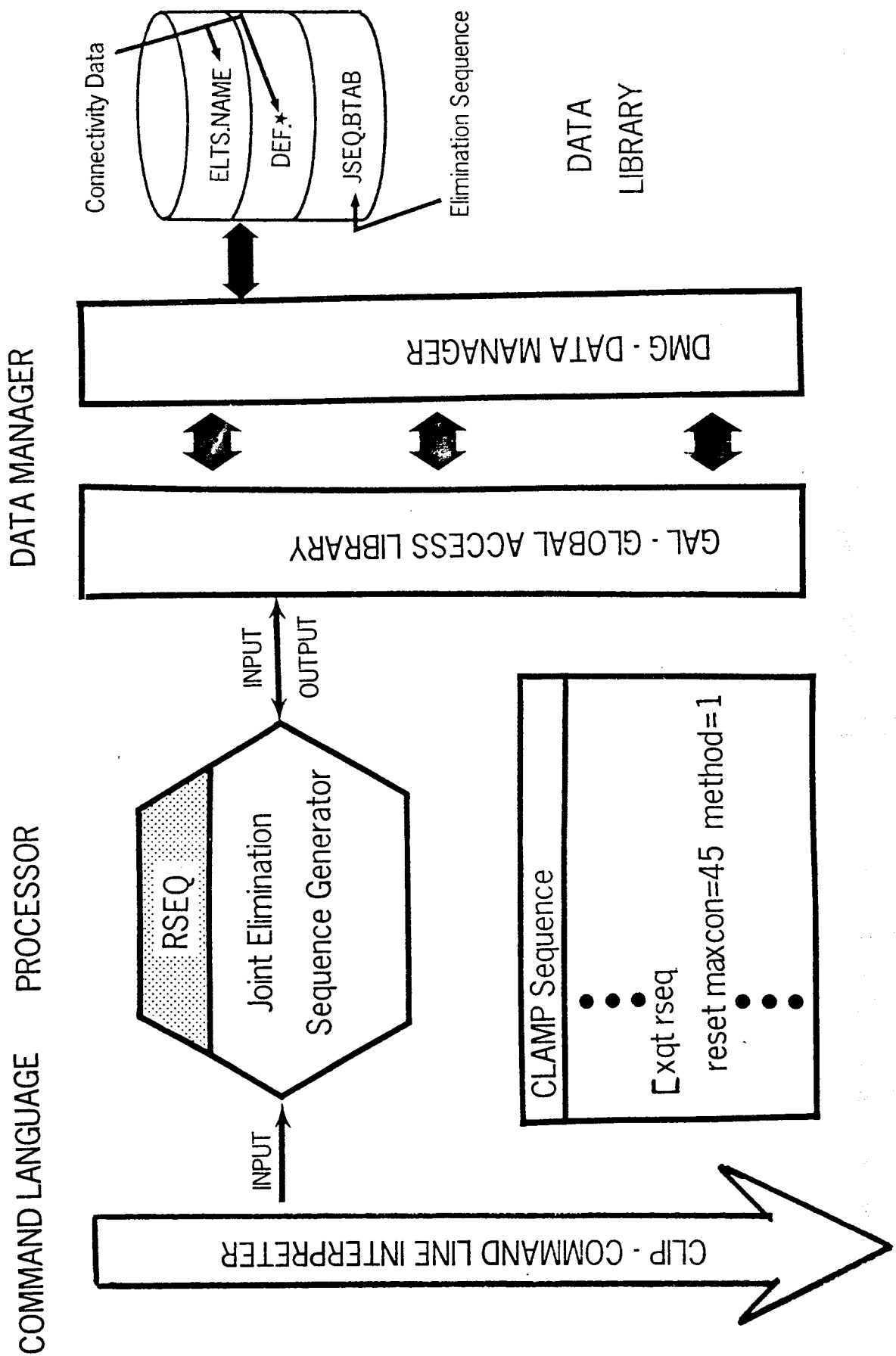
PROCESSOR DESCRIPTION

TAB	Basic Table Input
CSM1	Focus Problem Mesh Generation
LAU	Laminate Analysis Utility
ELD	Element Definition Processor
AUS	Arithmetic Utility System
RSEQ	Renumbering Strategies
E	E-State Initiation
TOPO	Element Topology Analyzer
K	System Stiffness Matrix Assembler
INV	SPAR Format Matrix Decomposition
SSOL	Static Solution Generation
GSF	Stress Data Generation
PSF	Stress Table Printer
KG	Geometric Stiffness Matrix Assembler
EIG	Sparse Matrix Eigensolver

PROCESSOR DESCRIPTION

The processors required to complete the buckling analysis are identified. These processors are controlled through the command language to produce the desired output for the analysis described earlier. Data is shared among processors only through the database.

PROCESSOR-ARCHITECTURE INTERACTION



PROCESSOR-ARCHITECTURE INTERACTION

Looking down one level from the buckling analysis level of Testbed application to the processor level we see the basic components of a modern structural analysis program. The processor, an independent analysis capability, is called into execution by the command language interpreter (CLIP). For example the functional processor RSEQ is initiated through the CLAMP(Command Language for Applied Mechanics) command "[xqt rseq". The processor uses CLIP input processing utilities to read and interpret the input parameters maxcon and method in order to correctly reset the default parameters.

RSEQ then reads the connectivity data from the data library through the global access library data base manager, GAL. GAL in turn uses a lower level I/O manager DMG which interfaces with the lower level "C" or assembly language routines to efficiently perform the I/O functions. This multi-level data base manager provides for an efficient machine independent capability by isolating machine dependent functions to the lowest level possible, and provides the most efficient machine interface possible at that level.

The output of the RSEQ processor is the dataset JSEQ.BTAB. This elimination sequence dataset determines the order of joint elimination in the solution phase based on the user specified method and can result in significant savings.

Each processor can be modified or replaced independently provided that the output data that would be created by that processor is the same as if it were created through another means.

CLIP COMMAND INTERPRETER

Function

- Execute CLAMP Directives to Control Processor Execution
- Provide a Data Description Language for Processor Input

CLAMP Directives

- Interface with the Global Data Manager
(*OPEN, *CLOSE, *TOC)
- Provide Command Procedure Management
(*PROCEDURE, *CALL)
- Command Processing Sequence Control
(*IF, *DO, *JUMP, *RETURN)
- Macrosymbol Directives
(*DEFINE, *SHOW MACRO, *UNDEFINE)
- SuperClip Directives
(*RUN, *STOP)
- Data Transfer Directives
(*LOAD, *UNLOAD)
- General Directives
(*HELP, *SHOW, *SET, *ADD, *DUMP)

CLIP COMMAND INTERPRETER

The CLIP command interpreter provides two basic functions for the processor. It controls the flow of input records to the executing processor via CLAMP directives, and it provides utilities to expand, construct, and parse processor input.

The CLAMP directives are broken down into seven general areas:

1. Interface with the Global Data Manager
2. Provide command procedure management
3. Command processing sequence control
4. Macro symbol directives
5. Superclip directives
6. Data transfer directives
7. General directives

TESTBED DATA MANAGEMENT

Features

- Global Data Libraries Contain Data Sets
- Data Sets are Accessed by Name
- Low Level I/O is Very Machine Dependent
- Data Storage is Addressed by Word
- Implementation is Hierarchical

Problems

- Data Set Names are Accessed Sequentially
- Management of Memory Resident Data

TESTBED DATA MANAGEMENT

Testbed data management is based on a hierarchic organization with datasets grouped into global data libraries which are currently implemented as files. Within the global data libraries (files) datasets are accessed by name by processors. Efficiency is important due to the size of the datasets that are involved in a state-of-the-art structural problem, and for this reason we resort to low level I/O that is written in assembly language, or "C" and is very machine dependent. We implement a word addressable scheme for managing data to simplify data address calculations. This results in a data handling scheme that provides the necessary efficiency without sacrificing simplicity.

The current implementation of the data management system is not without problems. For example, dataset names are stored in a directory structure called a TOC (table of contents) and are searched sequentially. This has resulted in excessive time when the number of datasets gets large. We also maintain all data in the data management system in the form of files. We do not allow for memory resident data. In this day of computers with large memories, if we could avoid using disk we would decrease the time required for an analysis. We are working to resolve both of these problems at the current time and some partial results will be shown.

TESTBED ARCHITECTURE IMPROVEMENTS

- DATA MANAGER

PERFORMANCE ON TRANSIENT RESPONSE EXAMPLE

<u>VERSION</u>	<u>CPU SEC.</u>
1986	1120
1987	462

- COMMAND LANGUAGE

PERFORMANCE ON 10000 TRIP LOOP

<u>VERSION</u>	<u>CPU SEC.</u>
1986	681
1987	436

TESTBED ARCHITECTURE IMPROVEMENTS

Projects are underway in the two areas of executive control, data management and control language. The overhead of the sequential TOC search was eliminated through the use of a hash table and embedded linked lists. This resulted in a speedup of 2 1/2 for a transient response example problem. CLIP, on the other hand, presents a more difficult challenge. In simple analysis procedures the overhead presented by CLIP is not extreme; however, as the complexity of the procedures increases, we see a disproportionate increase in the CLIP overhead. Preliminary work on CLIP has resulted in a decrease in execution overhead but more work will have to be done in this area if we are to provide the ability to develop complex algorithms entirely within the control language.

CSM TESTBED SOURCE CODE CONTROL (CSM uVAX/ULTRIX)

- **MASTER SOURCE CODE (MSC) FILES FOR NICE AND SPAR**

Multiple machine versions in single file

VAX/VMS, VAX/ULTRIX, CRAY/UNICOS, SUN/UNIX

- **"TOOLS" for extracting compatible source code for target systems
MAX, INCLUDE**
- **"MAKEFILES" for building executable program on target systems**
- **Procedures for distributing latest version of MSC, TOOLS, and
MAKEFILES for target systems**
- **RCS (Revision Control System)**

UNIX utility for maintaining history of modifications to testbed code

CSM TESTBED SOURCE CODE CONTROL

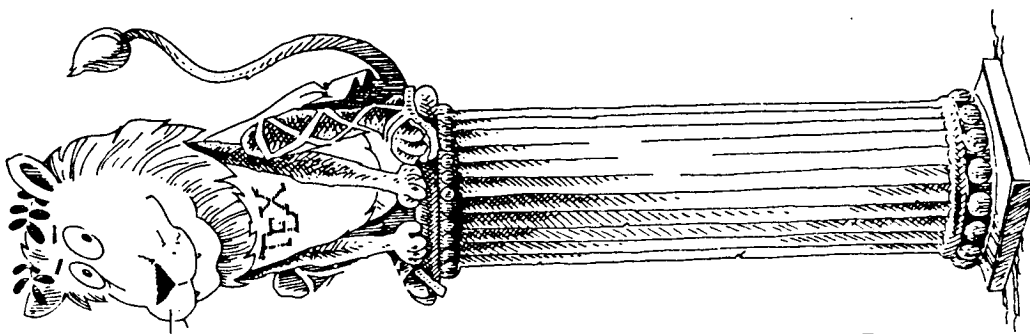
The CSM Testbed is available for many different computers and many different operating systems. If we are to be able to move the technology represented by the Testbed rapidly to a new architecture computer we must have efficient procedures to work with the source code. To accomplish this a group of "tools" has been assembled. We use the pre-compilers developed by Carlos Felippa, MAX and INCLUDE, to provide for extracting compatible source code for target machines. We use makefiles for building executable programs on target machines. We use the Revision Control System (RCS) provided with the UNIX operating system to maintain a history of modifications to the master source code. In addition, we have developed distribution procedures that allow for rapid system updates.

If we are to have a number of simultaneous developers we must maintain control of the Testbed source code and by maintaining strict control of the source code development we are able to transfer new capabilities faster.

CSM Testbed Documentation Set

Proposed Manuals

1. User's Manual (50%)
2. Theory Manual (5%)
3. Demo Manual (updated as available)
4. Programmer's Manual (0%)
5. Architecture Manuals (NASA CR, 5 volumes) (99%)
6. Data Library Description (90%)
7. Introduction (NASA TM 89096)
8. Documentation and Programming Standards Manual (0%)
9. Software Tools (NASA CR) (99%)

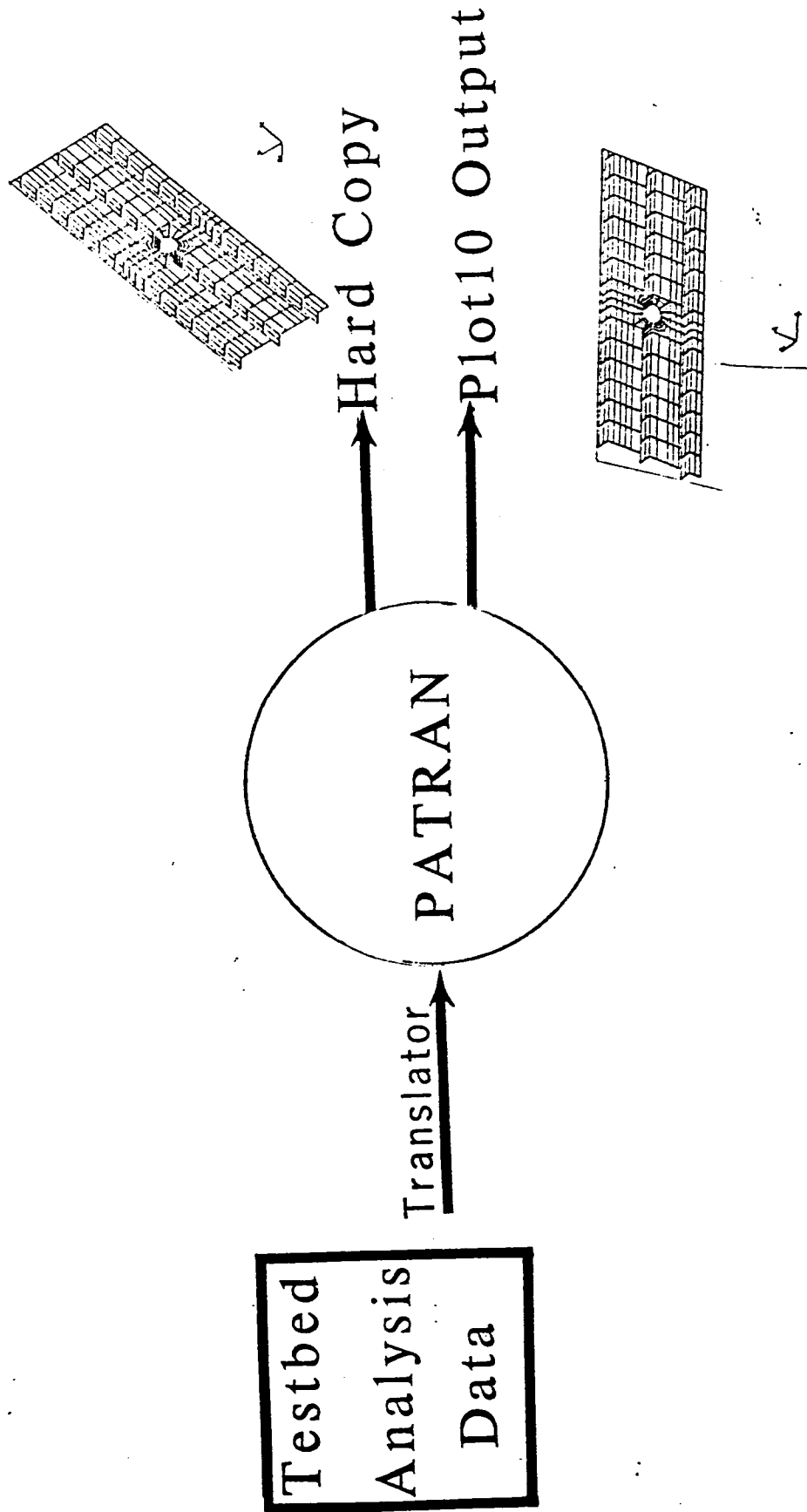


CSM TESTBED DOCUMENTATION SET

The nine manuals proposed in this documentation set are a priority item for the CSM Testbed development team. We are collecting and publishing material through NASA Contractor Reports as well as NASA Technical Memorandums. All documentation will be prepared using the T_EX documentation formatter in order to make updates easier.

The Introduction to the CSM Testbed has been released as NASA TM 89096. The 5-volume NICE Architecture manuals are almost ready for release as a NASA CR. In-house priority is being given to the User's Manual.

TESTBED GRAPHICS POSTPROCESSING



TESTBED GRAPHICS POSTPROCESSING

We are using the PATRAN system to provide output postprocessing for the CSM Testbed. In order to interface this product with the Testbed a translator was developed to convert the output of the Testbed for use by PATRAN. In addition, the ability to get simple line drawings out of PATRAN required an interface for PATRAN to be developed to allow a file compatible with the Tektronix PLOT10 output to be generated. These two translators were developed in-house and are now available. Hard copy output from PATRAN can now be either through the hard copy device on the graphics display device, or through simple line drawings output and processed through the PLOT10 translator.

Coupling PATRAN with the Testbed gives us the ability to delay a decision on the in-house development of a graphics code. At the present we are concentrating our effort in the development of code that would actually perform the structural analysis in a truly dynamic computational environment. We will review the decision on graphics at a later time.

TESTBED DEVELOPMENT TIMETABLE

1987

- CLEAN UP EXISTING TESTBED
- IMPROVE EFFICIENCY AND EXTENDABILITY
- ADD MODEST NUMBER OF NEW CAPABILITIES
- IMPROVE DOCUMENTATION
- DEVELOP COMPUTER INDEPENDENT CODE MANAGEMENT TECHNIQUES
- IMPLEMENT GRAPHICS POSTPROCESSING (PATRAN)

1988

- IMPLEMENT GRAPHICS PREPROCESSING (PATRAN)
- USE FOR EVALUATING STRUCTURAL ANALYSIS AND COMPUTATIONAL METHODS, DATA HANDLING TECHNIQUES, HIGHER-ORDER LANGUAGES, AND ADVANCED ARCHITECTURE CONCEPTS
- BEGIN TO MODIFY ARCHITECTURE TO EXPLOIT PARALLELISM
- USE FOR SOLVING CHALLENGING PROBLEMS

1989

- MOVE AGGRESSIVELY TOWARD ARCHITECTURE FOR EXPLOITING PARALLEL COMPUTERS
- PROVIDE POWERFUL, EASY-TO-USE TESTBED

TESTBED TIMETABLE

The timetable for the CSM Testbed group is presented for the last year, the next year, and through 1989.

In the last year we concentrated on making the existing Testbed more usable. We put a large effort in cleaning up the Testbed and improving the efficiency. We extended the computer environment onto new processors including high end workstations and supercomputers. We added a modest number of new capabilities to the existing SPAR processors and the NICE architecture. We have improved and continue to improve the documentation to allow a growing number of developers to have access to the Testbed. We have developed a comprehensive set of computer independent code management techniques that permit rapid growth while maintaining tight control of the source code. And we have implemented graphic postprocessing through PATRAN.

In the next year we will continue with the items from last year with a stronger emphasis on problem solving. We will implement a graphics preprocessing interface to PATRAN that will provide a greatly improved user interface. We also expect this preprocessor will assist in converting NASTRAN models for use in the Testbed, for example. We intend to use higher level language development tools to produce an improved command language for the Testbed. We will also begin to modify the software architecture to exploit the new parallel computers. And above all we will prove the Testbed's capability for solving new, challenging problems.

As we go farther out two years to 1989 we get more vague about the tactics of Testbed development and concentrate on the overall strategy. We will move toward a software architecture that will exploit powerful parallel computers while continuing to provide a powerful, easy-to-use Testbed.

TESTBED SCHEDULE

(Near Term - Next 12 months)

- COMPLETE DOCUMENTATION CURRENTLY IN PROGRESS
- INSTALL NEW PROCESSOR CAPABILITIES
- INSTALL ARCHITECTURE ENHANCEMENTS
- EXTEND SUPERCLIP TO UNIX
- REPLACE CLIP PARSER WITH IN-HOUSE DEVELOPMENT
- DEVELOP PATRAN PREPROCESSOR
- EXTEND PARALLEL ALGORITHM DEVELOPMENT TO NAS

TESTBED SCHEDULE

Looking at the next year we are expecting to see several milestones that represent real accomplishments in enhancing the usability of the Testbed.

1. Completing the documentation in progress. If the Testbed is to be usable by others besides the core CSM group, we must have good documentation available. The current documentation is usable by our in-house team.
2. New processors are being developed in-house, under the Lockheed contract, and as a result of grants. The Testbed group will be the clearing house for implementing the processors, testing their integration, and distributing them to others involved in the CSM activity.
3. Software Architecture enhancements are being developed in-house and under the Lockheed contract at the present time. The Testbed group will integrate the new enhancements into the Testbed and verify correct operation of a set of representative demonstration problems.
4. NICE, as originally envisioned by its developer Carlos Felippa, was considered to be a "Network of Independent Computational Elements", since such processors would be independently executed programs linked together by a loosely coupled executive system. This Superclick environment as it is called is currently available only on the VAX under VMS. Within the UNIX environment the Testbed is currently implemented as a single executable file that has all available processors linked into a large macro-processor environment. We intend to implement the VAX type of Superclick environment under UNIX as a first step toward adding the constructs that will permit parallel execution of the Testbed.
5. The largest single effort the Testbed group has undertaken for the upcoming year is the replacement of the NICE parser with an in-house development. The new parser will use the UNIX based utilities LEX and YAC to replace the ad-hoc generated parser in NICE. This will allow us to add new constructs to the command language quickly and easily through these UNIX table driven utilities.
6. A PATRAN preprocessor will be developed during the next year to provide the much needed user interface.
7. Many algorithms have been developed in-house by the parallel structural methods group. These algorithms for the most part have been developed for a FLEX-32 that the CSM group owns. The Testbed group plans to take the most promising algorithm produced by the parallel group and implement it on the CRAY-2 at Ames and evaluate the results.

CSM TESTBED SUMMARY

DEFINITION

A computer program for developing and evaluating advanced analysis methods and software architecture for a new generation of computers.

STATUS

- Work is progressing according to schedule
- Updated computer code is ready for release 1.1
- Documentation has reached the usable stage
- Enhancements are underway

CSM TESTBED SUMMARY

The CSM Testbed as it now exists is a computer program for developing and evaluating advanced analysis methods and software architecture for a new generation of computers. It works now, and the many enhancements planned provide the needed framework for research in this area.

Work is continuing according to an established schedule and will result in a capability that will advance the state-of-the-art.

The updated Testbed code is ready for release 1.1 with release 1.2 enhancements scheduled in about 3 months.

The documentation has reached the usable stage and will provide the first external documentation through the User's Manual within the next 4 months.

New capabilities are underway that will lead us through the next generation of computers. We intend to make this capability unique in the CSM research environment.